Distributed Communications Resource Management for Tracking and Surveillance Networks

Edwin K. P. Chong[a] and Brian Brewington[b]

[a]Colorado State University, 1373 Campus Delivery, Fort Collins, CO 80523, USA.
[b]Numerica Corporation, P. O. Box 271246, Fort Collins, CO 80527, USA.

ABSTRACT

We present a market-like method for distributed communications resource management in the context of networked tracking and surveillance systems. This method divides communication resources according to the expected utility provided by information of particular types. By formulating the problem as an optimization of the joint utility of information flow rates, the dual of the problem can be understood to provide a price for particular routes. Distributed rate control can be accomplished using primal-dual iteration in combination with communication of these route prices. We extend the previous work on the subject in a few important ways. First, we consider utility functions that are jointly-dependent on flow rates, to properly account for geometric synergy that can occur in sensor fusion problems. Second, we do not require that the rate-update algorithms have explicit knowledge of utility functions. Instead, our update algorithms involve transmitting marginal utility values. We present simulation results to demonstrate the effectiveness of the technique.

Keywords: Sensor management, communication resource management, tracking, surveillance

1. INTRODUCTION

Tracking and surveillance networks frequently suffer from significant communication resource constraints.[5] In the face of limited bandwidth and restrictions introduced by network topology, questions arise as to what information should be transmitted. This resource management problem is made more interesting by the existence of geometrically synergistic combinations of sensors that can best observe a target—that is, produce a given quality state estimate with minimal communication. For example, in video applications, the range to a target generally cannot be determined using a single observer (absent ground constraints or other information). A similar situation often exists with radar systems, in which case very accurate range information is available, but angular errors can be quite large. Networked tracking systems with sensors that lack quality information in some dimension benefit greatly from multiple observers, gaining the geometric diversity needed to form good quality tracks.

The problem then becomes one of determining the group of sensors and their relative contributions that provides the best information on a target. To this end, one can define a joint "utility" function that provides a measure of effectiveness as a function of the rate at which information of different types is received.[3] For example, an "information type" might be defined to be observations corresponding to a particular sensor-target pair, or perhaps a search for new targets by a particular sensor. This joint utility function and its derivatives are evaluated at certain points in the network at which the flows are collected, such as command centers.

The resulting problem has a simple form: maximize the (generally nonlinear) joint utility subject to bandwidth and flow constraints. Global planning methods for solving this problem can suffer when used in rapidly-changing environments—objective functions or constraints may change too quickly for a centralized solver to adapt and communicate a new strategy. A distributed solution makes sense: as local conditions or commander needs change, communications can be adapted so as to best utilize the available bandwidth.

In this work we present such a solution, which builds on an approach that has been widely applied in the communication-networks literature. The dual of the optimization problem (namely, joint utility maximization subject to capacity constraints) yields to an economic interpretation wherein each communication link has a price given by the dual variable in the Lagrangian associated with that link's capacity constraint. The route taken by a particular type of information, then, has a price equal to the sum of the prices of the links along the route. Communication of these route

---

E-mail: echong@colostate.edu, bebrewington@numerica.us

		Form Approved
Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **AUG 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **Distributed Communications Resource Management for Tracking and Surveillance Networks**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Colorado State University,1373 Campus Delivery,Fort Collins,CO,80523**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets, San Diego, CA August 2005

14. ABSTRACT

We present a market-like method for distributed communications resource management in the context of networked tracking and surveillance systems. This method divides communication resources according to the expected utility provided by information of particular types. By formulating the problem as an optimization of the joint utility of information flow rates, the dual of the problem can be understood to provide a price for particular routes. Distributed rate control can be accomplished using primal-dual iteration in combination with communication of these route prices. We extend the previous work on the subject in a few important ways. First, we consider utility functions that are jointly-dependent on flow rates, to properly account for geometric synergy that can occur in sensor fusion problems. Second, we do not require that the rate-update algorithms have explicit knowledge of utility functions. Instead, our update algorithms involve transmitting marginal utility values. We present simulation results to demonstrate the effectiveness of the technique.

15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

and link prices, along with the marginal utility of an information type, permits a naturally distributed optimization strategy in which sensors locally modify transmission rates. By including joint utility functions, we can account for the kind of geometric synergy mentioned earlier.

This paper is organized as follows: Section 2 discusses the utility and pricing structure, Section 3 covers the distributed rate control algorithm, and Section 4 shows some simple experimental verification of the algorithm.

2. UTILITY AND PRICING FRAMEWORK

2.1. Problem Formulation

The problem formulation here builds on the work of Brewington et al.[3] However, we deviate here in one aspect of the formulation: we assume that associated with each end-to-end communication is a predefined *route* (path) through the network (in Section 2.3 we discuss the case of multiple routes). This assumption is common in the literature on communication networks (see, e.g., Refs. 7–11, 13). In particular, we adopt a framework based on the work of Low and Lapsley,[9] which builds on a classical idea from economics.

In this section, we first provide a detailed formulation of the rate optimization problem. We then discuss a distributed solution approach to the rate optimization problem, based on duality theory.

2.1.1. Network and data types

To begin, consider a communication network consisting of a set of nodes and a set of L links connecting these nodes. Index the links by $l = 1, \ldots, L$. Associated with each link l is a fixed capacity value c_l.

We have N *data types* to be communicated over the network. Each data type has a source node and a destination node. Index the data types by $i = 1, 2, \ldots, N$. For each data type, we refer to the data communicated through the network as its *flow*.

EXAMPLE 1. (Motivated by the setup in Brewington et al.[3]) Consider a set of sensors Σ and a set of targets T (capital "tau"). Each sensor is located at a node, and has measurements from one or more of these targets. For each sensor, these target measurements are to be sent from the sensor node to a commander (located at some other node). In this case, each sensor-target pair (σ, τ), $\sigma \in \Sigma$, $\tau \in T$, is a data type. □

Associated with each data type $i = 1, \ldots, N$ is a prespecified *route*, which is an acyclic list of consecutively conjoined links through the network starting at the source of data type i and ending at its destination (the route is fixed but otherwise arbitrary). Let $R(i)$ represent the set of links on the route of data type i. For data type i, the quantity of flow being communicated is called the *flow rate* (a nonnegative quantity), denoted f_i. Associated with each flow rate are prespecified maximum and minimum constraints: $m_i \leq f_i \leq M_i$.

Each link in the network may carry flows from multiple data types. For each link $l = 1, \ldots, L$, let $F(l)$ be the set of data types communicating through it. The sum of the flow rates on each link should not exceed the link capacity: $\sum_{i \in F(l)} f_i \leq c_l, l = 1, \ldots, L$.

2.1.2. Utility optimization

Our task is to set the values of the flow rates $f_1, \ldots, f_N$ (subject to the link-capacity constraints mentioned above). The setting of these values is based on maximizing a *joint utility function* $U(f_1, \ldots, f_N)$. This notion of a joint utility function is more general than the utility formulation in Brewington et al.[3] and in Low and Lapsley.[9]

For convenience, let $\boldsymbol{f} = [f_1, \ldots, f_N]^T$ denote the vector of flow rates, and (with some harmless abuse of notation) let $U(\boldsymbol{f})$ be the (joint) utility value associated with $\boldsymbol{f}$. Let $\mathcal{D} = \{\boldsymbol{f} : m_i \leq f_i \leq M_i, \ i = 1, \ldots, N\}$ denote the set of rate vectors satisfying the max and min constraints on their respective components. We call any such vector *feasible*.

To summarize, our optimization problem is to find a feasible rate vector maximizing the joint utility function subject to link-capacity constraints:

$$\begin{aligned} &\underset{\boldsymbol{f} \in \mathcal{D}}{\text{maximize}} \ U(\boldsymbol{f}) \\ &\text{subject to} \ \sum_{i \in F(l)} f_i \leq c_l, \quad l = 1, \ldots, L. \end{aligned} \tag{1}$$

EXAMPLE 2. Consider the sensor-target data-type scenario from Example 1. Suppose that the utility functions are jointly dependent only through the sensors that provide the data for that target, but are not jointly dependent across targets. To elaborate, for each target $\tau \in T$, let U_τ be the utility function for target τ, and $\boldsymbol{f}_\tau$ the vector of flows associated with τ (the components of $\boldsymbol{f}_\tau$ correspond to all sensor-target pairs (σ, τ), $\sigma \in \Sigma$). The utility function U_τ depends only on $\boldsymbol{f}_\tau$, and not on any other flow rates. Then, following Brewington et al.,[3] the overall joint utility function is given by $U(\boldsymbol{f}) = \sum_{\tau \in T} U_\tau(\boldsymbol{f}_\tau)$.

A further special case is where U_τ depends on $\boldsymbol{f}_\tau$ only through the aggregate flow rate for target τ; i.e., there is a function $\bar{U}$ such that $U_\tau(\boldsymbol{f}_\tau) = \bar{U}\left(\sum_{\sigma \in \Sigma} f_{(\sigma, \tau)}\right)$. □

EXAMPLE 3. Consider again the sensor-target data-type example from before. Suppose that we have number of commanders, indexed by $\kappa \in K$. Commander κ needs to receive only those flows that are associated with a certain subset of targets, T_κ. Let $\boldsymbol{f}_\kappa$ denote the vector of flows associated with T_κ (the components of $\boldsymbol{f}_\kappa$ correspond to all sensor-target pairs (σ, τ) such that $\tau \in T_\kappa$). Note that even for two distinct commanders κ_1 and κ_2, the vectors $\boldsymbol{f}_{\kappa_1}$ and $\boldsymbol{f}_{\kappa_2}$ may share one or more components.

Each commander has a utility function $U_\kappa(\boldsymbol{f}_\kappa)$ (which depends only on $\boldsymbol{f}_\kappa$ and not on the flows of other sensor-target pairs). Then, the overall joint utility function is $U(\boldsymbol{f}) = \sum_{\kappa \in K} U_\kappa(\boldsymbol{f}_\kappa)$.

The utility functions U_κ may encode information about the relative "values" (or "ranks") of different commanders. For example, it may be the case that

$$\max_{\boldsymbol{f}_{\kappa_1}} U_{\kappa_1}(\boldsymbol{f}_{\kappa_1}) > \max_{\boldsymbol{f}_{\kappa_2}} U_{\kappa_2}(\boldsymbol{f}_{\kappa_2}),$$

reflecting a higher importance in maximizing the utility of commander κ_1. □

2.2. Types of Joint Utility Functions

In our framework, the utility function U depends jointly on the flow rates $f_1, \ldots, f_N$. It is useful to categorize different ways in which the utility depends jointly on these flow rates. In the simplest case, the joint utility function is the sum of utility functions of individual flows:

$$U(f_1, \ldots, f_N) = U_1(f_1) + \cdots + U_N(f_N).$$

We refer to such flows as *independent*. This is the usual case considered in the data-networking literature (e.g., Refs. 7–11, 13). It is easy to show that if each individual U_i, $i = 1, \ldots, N$, is concave, then U is concave.

A second category of joint dependence is characterized by the following:

$$U(f_1, \ldots, f_N) = U_c(\min(f_1, \ldots, f_N))$$

for some $U_c : \mathbb{R} \to \mathbb{R}$. We refer to such flows as *complementary*, following the classical terminology in the economics literature (e.g., Ref. 12). The basic idea here is that the utility value depends on a "combination" of flows, so that if we only have a certain amount of data from one flow, no additional utility is gained by having more data from other flows. It is easy to show that if U_c is concave, then so is U.

The above form of utility function U reflects an extreme form of complementary flows, often called "perfect" complements (analogous to nuts and bolts). Less extreme forms of complementarity (more like bread and butter) involve *some* gain in utility from data of flows greater than the least flow, but not as much as would be obtained in the independent case. Complementary flows are particularly relevant in tracking/surveillance applications. For example, tracking performance typically depends on a combination of data from multiple sensors, and the lack of data from one sensor cannot be significantly ameliorated by more data from other sensors. This is common in tracking when information on some dimension is poor or even totally unavailable (as can be true in radar or visible-band surveillance).

A third category of joint dependence is given by the following:

$$U(f_1, \ldots, f_N) = U_s(f_1 + \cdots + f_N)$$

for some $U_s : \mathbb{R} \to \mathbb{R}$. In this case, the flows are called *substitute* flows—the data from one flow can be substituted by data from other flows with no change in utility. The above form of utility function involves "perfect" substitution. Less extreme
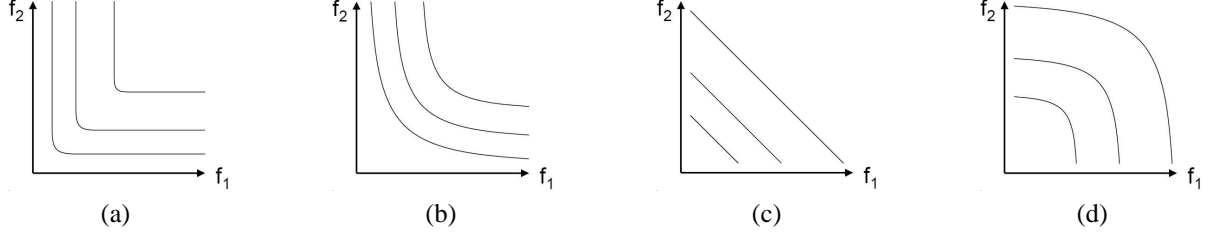
Figure 1. Level sets of utility functions with two flows: (a) complementary (concave); (b) independent (concave); (c) substitute (concave); (d) duplicate (not concave).

forms of substitute flows involve *some* change in utility if we substitute one flow with another. It is easy to show that if U_s is concave, then so is U.

In general, utility functions could involve combinations of flow types. For example: $U(f_1, f_2, f_3) = U_1(f_1) + U_{2,3}(f_2, f_3)$ where f_2 and f_3 are complementary flows, and f_1 is independent of the pair (f_2, f_3). This example is considered in Section 4.

One other category of flows is worth mentioning: *duplicate* flows. In the "perfect" duplicate case, we have

$$U(f_1, \ldots, f_N) = U_d(\max(f_1, \ldots, f_N))$$

for some $U_d : \mathbb{R} \to \mathbb{R}$. In this case, information from one flow duplicates information from another—once we have data from one flow, no additional utility is gained from other lesser flows. This would be the case if, for example, we send the same sensor data over multiple routes, treating each route as a distinct flow. We typically do not consider the duplicate-flow case because the utility function U in this case is not concave, even if U_d is concave. Concavity is important in solving the utility maximization problem; see Section 3.1.

Figure 1 illustrates the types joint utility functions discussed above.

2.3. Multiple Routes per Flow

Our formulation thus far of the rate-control problem assumes a single route for each data type. This assumption is common in the current literature on data-communication networks. However, it is often of interest to consider the case where each data type has multiple routes for communicating its data. It turns out that by having joint utility functions, the single-route assumption holds without loss of generality. In particular, we show here how to account for the case where a single data type has multiple routes.

Consider a particular data type i and a set of multiple routes, indexed by $k = 1, \ldots, R$, associated with this data type. Suppose that the flow for data type i is *split* across these routes, so that the aggregate flow over these routes constitutes the flow for the data type, denoted f_i as usual.

For each k, define a *virtual* data type, indexed by the pair i, k, and associate with this virtual data type the route k. Let $f_{i,k}$ be the flow rate for virtual data type i, k. Then, the flow rate for data type i is given by $f_i = f_{i,1} + \cdots + f_{i,R}$. We can now treat each virtual data type as an individual data type in the setting of our rate-control problem, where replace data type i by the virtual data types. Specifically, the utility function U (which has f_i as one of its arguments) is replaced by the following:

$$U_v(\cdots, f_{i,1}, \ldots, f_{i,R}, \cdots) = U(\cdots, f_{i,1} + \cdots + f_{i,R}, \cdots).$$

In other words, the constituent (virtual) flows for data type i are *perfect substitute* flows with respect to the joint utility function involving the virtual data types. Our previous framework (assuming a single route per flow) now applies.

2.4. Utility on Object Types

Suppose we have a set of object types and a utility function over the quantities of objects. Specifically, if $j = 1, \ldots, M$ is the index for object types, and $g_1, \ldots, g_M$ are their (scalar) "quantities," then $V(g_1, \ldots, g_M)$ is the utility received. Our goal is to maximize this utility value. But we do not have direct control over $g_1, \ldots, g_M$. What we can control are the data-type flow rates $f_1, \ldots, f_N$, and these flow rates affect the object-type quantities. In a surveillance application, this

might manifest itself as obtaining benefit from observations of a certain type of target, where one can only control the rate at which observations of all targets are sent. Similar arguments can be made to handle observation-to-track association ambiguity; there may be utility only for correct associations.

To capture how the flow rates affect the object quantities, we use the following model. For each data type, we have a probability distribution over object types. (Call this the *data association law*.) This allows us to construct a utility function over the flow rates, $U(f_1, \ldots, f_N)$, such that maximizing this utility function maximizes the *mean* utility value of object quantities. The framework we have developed above is therefore also useful in this context.

To be precise, let $\boldsymbol{f}$ and $\boldsymbol{g}$ be vectors representing data-type flow rates and object-type quantities, respectively. Then, given $\boldsymbol{f}$, the data association law is specified by a conditional probability distribution $P(\cdot|\boldsymbol{f})$ over possible values of object-type quantities. Given the utility function V over object-type quantities, define a utility function U over data-type flow rates by

$$U(\boldsymbol{f}) = E[V(\boldsymbol{Q})|\boldsymbol{f}] = \sum_{\boldsymbol{g}} V(\boldsymbol{g})P(\boldsymbol{g}|\boldsymbol{f}). \tag{2}$$

Then, maximizing $U(\boldsymbol{f})$ (under the framework introduced earlier) maximizes the conditional mean of $V(\boldsymbol{g})$ given $\boldsymbol{f}$ (with respect to the data association law).

EXAMPLE 4. Consider once again the scenario with sensors taking measurements of targets. Suppose that each sensor has a number of measurements, but we cannot be certain which measurement corresponds to which target. Instead, for each measurement, we have a probability distribution over targets—this distribution specifies, for each target, the probability that each given measurement is associated with that target.

Suppose object types represent targets, and data types represent sensor-measurement pairs. So f_i represents the flow rate for sensor-measurement pair i, and g_τ represents the measurement rate for target τ. What a commander specifies is (as before) a joint utility function V over rates of measurements for targets $V(\boldsymbol{g})$.

As a special case, the commander may specify, for each target $\tau \in T$, the utility function $V_\tau(g_\tau)$ (which depends only on the measurement rate for target g_τ. The target-specific utility functions give rise to an overall utility function $V(\boldsymbol{g}) = \sum_{\tau \in T} V_\tau(g_\tau)$. We are also given the data association law $P(\boldsymbol{g}|\boldsymbol{f})$. We can now define $U(\boldsymbol{f})$ via (2) and apply the framework described above. □

3. DISTRIBUTED RATE-CONTROL ALGORITHM

3.1. Primal-Dual Formulation of Rate Control Problem

Our next goal is to develop a distributed scheme to solve the basic rate-control problem (1). We follow the duality approach of Low and Lapsley.[9] This approach has proven to be successful in dealing with distributed congestion control problems in data networks, including the Internet, with significant ongoing work (see, e.g., Refs. 9, 10, 13, and also Refs. 7, 8, 11 for some earlier work along related lines). For an excellent treatment of duality theory for nonlinear programming problems, see Boyd and Vandenberghe.[1]

The duality approach goes roughly as follows. For the given problem (1), we introduce another related problem called its "dual," with the property that solving the dual somehow aids in solving the primal. In fact, the dual and primal taken together yield an algorithm that has a natural distributed implementation.

To begin, define the Lagrangian function

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{p}) = U(\boldsymbol{f}) + \sum_{l=1}^{L} p_l \left(c_l - \sum_{i \in F(l)} f_i \right),$$

where $\boldsymbol{p} = [p_1, \ldots, p_L]^T$ is a vector of *dual variables* (often also called *shadow prices*, or simply *prices*). In the context of our problem, we will call these variables the *link prices*.

For convenience, the Lagrangian can be rewritten as

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{p}) = U(\boldsymbol{f}) + \sum_{l=1}^{L} p_l \left(c_l - \sum_{i \in F(l)} f_i \right) = U(\boldsymbol{f}) - \sum_{i=1}^{N} f_i r_i + \sum_{l=1}^{L} p_l c_l = U(\boldsymbol{f}) - \boldsymbol{r}^T \boldsymbol{f} + \boldsymbol{c}^T \boldsymbol{p},$$

where $r_i = \sum_{l \in R(i)} p_l$ represents the *route price* for the data type i, $\boldsymbol{r} = [r_1, \ldots, r_N]^T$ is the vector of route prices, and $\boldsymbol{c} = [c_1, \ldots, c_L]^T$ is the vector of link capacities.

Next, we construct the *dual* objective function:

$$D(\boldsymbol{p}) = \max_{\boldsymbol{f} \in \mathcal{D}} \mathcal{L}(\boldsymbol{f}, \boldsymbol{p}) = \max_{\boldsymbol{f} \in \mathcal{D}} \left(U(\boldsymbol{f}) - \sum_{i=1}^{N} f_i r_i \right) + \sum_{l=1}^{L} p_l c_l = \max_{\boldsymbol{f} \in \mathcal{D}} \left(U(\boldsymbol{f}) - \boldsymbol{r}^T \boldsymbol{f} \right) + \boldsymbol{c}^T \boldsymbol{p}.$$

Let us call (1) the *primal* problem. The corresponding *dual* problem is given by

$$\underset{\boldsymbol{p} \geq \boldsymbol{0}}{\text{minimize}} \ D(\boldsymbol{p}), \tag{3}$$

where the notation $\boldsymbol{p} \geq \boldsymbol{0}$ means that every component of $\boldsymbol{p}$ is nonnegative.

We make two additional assumptions (and continue to hold them for the remainder of the paper):

A1. The function U is concave.

A2. There exists $\boldsymbol{f}$ such that $m_i < f_i < M_i$, $i = 1, \ldots, N$, and $\sum_{i \in F(l)} f_i \leq c_l$, $\quad l = 1, \ldots, L$.

Assumption A1 is relatively standard in the theory of utility functions. Assumption A2, called Slater's condition, is natural in our context, and should be expected to hold in practice (it requires only that there is an *interior* point of $\mathcal{D}$ satisfying link-capacity constraints). Under these two assumptions, a condition called *strong duality* holds (see Ref. 1). Strong duality has the following consequences. If $\boldsymbol{p}$ is a solution to the dual problem, then $\arg\max_{\boldsymbol{f} \in \mathcal{D}} \mathcal{L}(\boldsymbol{f}, \boldsymbol{p})$ is a solution to the primal problem (i.e., is a vector of optimal rates). But

$$\arg\max_{\boldsymbol{f} \in \mathcal{D}} \mathcal{L}(\boldsymbol{f}, \boldsymbol{p}) = \arg\max_{\boldsymbol{f} \in \mathcal{D}} \left(U(\boldsymbol{f}) - \boldsymbol{r}^T \boldsymbol{f} \right).$$

Thus, once we have an optimal price vector $\boldsymbol{p}$, we can find the optimal flow rates by maximizing $U(\boldsymbol{f}) - \boldsymbol{r}^T \boldsymbol{f}$. Conversely, if $\boldsymbol{f}$ is an optimal rate vector, then (again under strong duality) $\arg\min_{\boldsymbol{p} \geq \boldsymbol{0}} \mathcal{L}(\boldsymbol{f}, \boldsymbol{p})$ is a solution to the dual problem. This primal-dual optimization result forms the basis for a distributed rate-control scheme.

3.2. Distributed Rate Control: Primal-Dual Iterations

We now describe a "primal-dual iteration" method to find the optimal prices and rates, following Low and Lapsley.[9] In other words, we iteratively update rates and prices, one at a time: for the primal we maximize the Lagrangian with respect to $\boldsymbol{f}$, and for the dual we minimize the Lagrangian with respect to $\boldsymbol{p}$. To be more precise, let t be the iteration index. We construct a scheme to update $(\boldsymbol{f}(t), \boldsymbol{p}(t))$ to $(\boldsymbol{f}(t+1), \boldsymbol{p}(t+1))$, in two steps, as follows. First, we update only the dual variable $\boldsymbol{p}(t)$, keeping $\boldsymbol{f}(t)$ fixed, to obtain $(\boldsymbol{f}(t), \boldsymbol{p}(t+1))$. Then, we update the primal variable $\boldsymbol{f}(t)$, keeping $\boldsymbol{p}(t+1)$ fixed, to obtain $(\boldsymbol{f}(t+1), \boldsymbol{p}(t+1))$. These steps are described next.

For the dual, we use a "projected gradient algorithm" (see, e.g., Ref. 4) to compute prices:

$$\boldsymbol{p}(t+1) = [\boldsymbol{p}(t) - \gamma \nabla_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{f}(t), \boldsymbol{p}(t))]_+,$$

where $\nabla_{\boldsymbol{p}} \mathcal{L}$ is the gradient of $\mathcal{L}$ with respect to its price argument, $\gamma > 0$ is a (sufficiently small) step size, and $[\cdot]_+ = \max(\cdot, 0)$. Let $\nabla_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{f}(t), \boldsymbol{p}(t))$ represent the gradient of $\mathcal{L}$ with respect to its second argument, $\boldsymbol{p}$, evaluated at $(\boldsymbol{f}(t), \boldsymbol{p}(t))$. Now, the lth component of $\nabla_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{f}(t), \boldsymbol{p}(t))$ (partial derivative with respect to p_l) is given by

$$\partial_{p_l} \mathcal{L}(\boldsymbol{f}(t), \boldsymbol{p}(t)) = c_l - \sum_{i \in F(l)} f_i(t) = c_l - a_l(t)$$

where $a_l = \sum_{i \in F(l)} f_i$ is the *aggregate link rate* for link l. Hence, the projected gradient algorithm can be written component-wise (i.e., in a link-by-link fashion) as

$$p_l(t+1) = [p_l(t) + \gamma(a_l(t) - c_l)]_+, \quad l = 1, \ldots, L. \tag{4}$$

Notice that this algorithm relies only on information that is locally available at link l, and hence has a naturally distributed implementation. Note also that in the course of the algorithm we allow the aggregate link rate $a_l(t)$ to exceed the link capacity c_l, so that $a_l(t) - c_l$ may be positive. In this case, the algorithm increases the price at iteration t. The goal is that the algorithm converges to a point where the aggregate link rates satisfy the link-capacity constraints.

Having updated the dual variable $\boldsymbol{p}(t)$ to $\boldsymbol{p}(t+1)$, we now fix $\boldsymbol{p}(t+1)$ and update the primal variable $\boldsymbol{f}(t)$ to $\boldsymbol{f}(t+1)$. At iteration t, we have

$$\boldsymbol{f}(t+1) = \arg\max_{\boldsymbol{f} \in \mathcal{D}} \left(U(\boldsymbol{f}) - \boldsymbol{r}(t+1)^T \boldsymbol{f} \right),$$

where we recall that $\boldsymbol{r}(t+1) = [r_1(t+1), \ldots, r_N(t+1)]^T$ is the vector of route prices at iteration t, and $r_i(t+1) = \sum_{l \in R(i)} p_l(t+1)$ is the route price for data type i (after the dual update). Assume that U is differentiable. Let $\hat{\boldsymbol{f}}(t+1)$ satisfy $\nabla U(\hat{\boldsymbol{f}}(t+1)) = \boldsymbol{r}(t+1)$. Then, for the ith component of $\boldsymbol{f}(t+1)$, set $f_i(t+1) = [\hat{f}_i(t+1)]_{m_i}^{M_i}$, where the notation $[\cdot]_{m_i}^{M_i}$ is defined by

$$[f]_{m_i}^{M_i} = \begin{cases} m_i & \text{if } f \leq m_i \\ f & \text{if } m_i < f < M_i \\ M_i & \text{if } f \geq m_i, \end{cases}$$

which represents the projection operator into the set of feasible rates.

We can implement the computation of $\boldsymbol{f}(t+1)$ in a distributed way. To describe such an implementation, suppose we have a number of *local updating agents*. We associate with each agent one or more data types—each agent performs the updates of the rate variables associated with it.

EXAMPLE 5. Suppose data types represent sensor-target pairs, and agents correspond to sensors. Then, each sensor σ updates the rates for all sensor-target pairs associated with it: $f_{\sigma\tau}, \tau \in T$. $\square$

For simplicity of notation, consider the agent associated with data types $1, \ldots, m$. This agent performs the following update: compute $\hat{f}_1(t+1), \ldots, \hat{f}_m(t+1)$ by solving the set of equations

$$\partial_{f_i} U(\hat{f}_1(t+1), \ldots, \hat{f}_m(t+1), f_{m+1}(t), \ldots, f_N(t)) = r_i(t+1), \quad i = 1, \ldots, m \qquad (5)$$

and then setting $f_i(t+1) = [\hat{f}_i(t+1)]_{m_i}^{M_i}$, $i = 1, \ldots, m$. This method of computing $f_i(t+1)$ is akin to "coordinate descent" (see, e.g., Ref. 2), although there may be multiple "coordinates" being updated simultaneously here. Notice that some communication between updating agents is necessary: to update its own component(s), every agent uses other components from the vector $\boldsymbol{f}(t)$. To be more precise, updating the components $1, \ldots, m$ involves knowledge of the marginal utility functions

$$\partial_{f_i} U(f_1, \ldots, f_m, f_{m+1}(t) \ldots, f_N(t)), \quad i = 1, \ldots, m$$

(as functions of $f_1, \ldots, f_m$). Information necessary to compute this marginal utility function needs to be communicated to the agent updating the rates for data types $1, \ldots, m$. We address this issue further in the next section.

3.3. Distributed Rate Control: Some Refinements

The rate-control method in the last section follows Low and Lapsley,[9] who assume that the utility function involves *independent* flows, and that the each individual flow source acts as the local updating agent for the flow. In the case of independent flows, the update equation (5) can be implemented at individual sources without any communication of rates between sources. This is not the case in our general formulation—if the flows are not independent, then (5) involves knowledge of potentially all N rate values. Because these values are usually not available locally at the updating agent, they have to be communicated. This communication may incur a prohibitive burden on network resources. Moreover, in many situations of interest the utility functions are unknown to the updating agents. For example, the utility functions might represent target tracking performance metrics, such as the trace of the error covariance.[3] Utility is likely best connected to a mission at some downstream recipient (say, a centralized tracker), while updating agents are typically implemented at sensor locations. Furthermore, because of the mobility of the target and the sensor platform, utility functions will vary with time.

To overcome the problem of requiring the updating agents to know the utility functions, and having to communicate rate values to implement (5), we introduce some modifications to the rate-update mechanism. First, we note that instead of

updating the rate vector using (5), we could use a projected gradient algorithm similar to (4). Specifically, the local update agent for flow i computes $\hat{f}_i(t+1)$ using

$$f_i(t+1) = [f_i(t) + \alpha(\partial_{f_i} U(\boldsymbol{f}(t)) - r_i(t+1))]_{m_i}^{M_i}. \tag{6}$$

This form of rate updating, like (4), is "incremental" in nature. The step size $\alpha > 0$ controls the size of the increment at each iteration. Note that implementation of (6) involves knowing only the *value* of the marginal utility $\partial_{f_i} U(\boldsymbol{f}(t))$; we do not need explicit knowledge of the rate vector $\boldsymbol{f}(t)$ or of the marginal utility *function*, in contrast to (5). The marginal utility values can be obtained (calculated) by the consumer of the data (e.g., a commander), who already receives the flows and does not need flow rates explicitly communicated to him. These marginal utility values still have to be communicated to the rate-updating agent, but this communication burden is potentially a small fraction of the burden involved in (5).

In practice, the implementation of the update mechanisms (4) and (6) are asynchronous and involves delayed versions of rates and prices. Specifically, in the right-hand side of (4), the aggregate link rate that is used is the *actual* link rate at the time t of the update, which typically involves delayed versions of flow rates (computed at times prior to t). Similarly, in the right-hand side of (6), the route price involves link prices available to the updating agent at time t, typically computed prior to time t, and the marginal utility value involves flow rates updated prior to time t. Such delays typically do no violence to the asymptotic properties of the algorithm.[2] However, the transient behavior may be affected significantly (e.g., large oscillations).

To improve the transient behavior of the price and rate update mechanisms, we introduce "second-order" terms to the update equations. Specifically, for the price-update algorithm for link l, we use

$$\delta_l(t) = (\text{flow on link } l \text{ at time } t) - c_l$$
$$p_l(t+1) = [p_l(t) + \gamma_1 \delta_l(t) + \gamma_2(\delta_l(t) - \delta_l(t-1))]_+,$$

and for the rate-update algorithm for flow i we use

$$\beta_i(t) = (\text{marginal utility at } t) - (\text{route price at } t)$$
$$f_i(t+1) = [f_i(t) + \alpha_1 \beta_i(t) + \alpha_2(\beta_i(t) - \beta_i(t-1))]_{m_i}^{M_i}.$$

These update algorithms are akin to "PI controllers," common in industrial control systems.[6] Note that there are now two step-size parameters in each algorithm, one for the first-order term (as before) and another for the second-order term. Appropriate setting of these parameter values improves the transient behavior significantly.

3.4. KKT Optimality Condition

Assuming A1 and A2 as before, the Karush-Kuhn-Tucker (KKT) condition provides a necessary and sufficient condition for optimality. To elaborate, suppose $(\boldsymbol{f}, \boldsymbol{p})$ is a given feasible pair of flow-rate and link-price vectors ($\boldsymbol{f} \in \mathcal{D}$ and $\boldsymbol{p} \geq \boldsymbol{0}$). Then, this pair is optimal in their respective primal/dual problems if and only if the following hold:

$$
\begin{aligned}
\text{for } i = 1, \ldots, N: \quad & \partial_{f_i} U(\boldsymbol{f}) - r_i \leq 0 \quad \text{if } f_i = m_i \\
\text{for } i = 1, \ldots, N: \quad & \partial_{f_i} U(\boldsymbol{f}) - r_i = 0 \quad \text{if } m_i < f_i < M_i \\
\text{for } i = 1, \ldots, N: \quad & \partial_{f_i} U(\boldsymbol{f}) - r_i \geq 0 \quad \text{if } f_i = M_i \\
\text{for } l = 1, \ldots, L: \quad & p_l(c_l - a_l) = 0 \\
\text{for } l = 1, \ldots, L: \quad & c_l - a_l \geq 0.
\end{aligned}
$$

The above set of equations and inequalities is called the Karush-Kuhn-Tucker (KKT) condition.[4] For convenience, we rewrite all inequalities as equations using the equivalence $x \geq 0 \Leftrightarrow [x]_- = \min(x, 0) = 0$:

$$
\begin{aligned}
\text{for } i = 1, \ldots, N: \quad & [\partial_{f_i} U(\boldsymbol{f}) - r_i]_+ = 0 \quad \text{if } f_i = m_i \\
\text{for } i = 1, \ldots, N: \quad & \partial_{f_i} U(\boldsymbol{f}) - r_i = 0 \quad \text{if } m_i < f_i < M_i \\
\text{for } i = 1, \ldots, N: \quad & [\partial_{f_i} U(\boldsymbol{f}) - r_i]_- = 0 \quad \text{if } f_i = M_i \\
\text{for } l = 1, \ldots, L: \quad & p_l(c_l - a_l) = 0 \\
\text{for } l = 1, \ldots, L: \quad & [c_l - a_l]_- = 0.
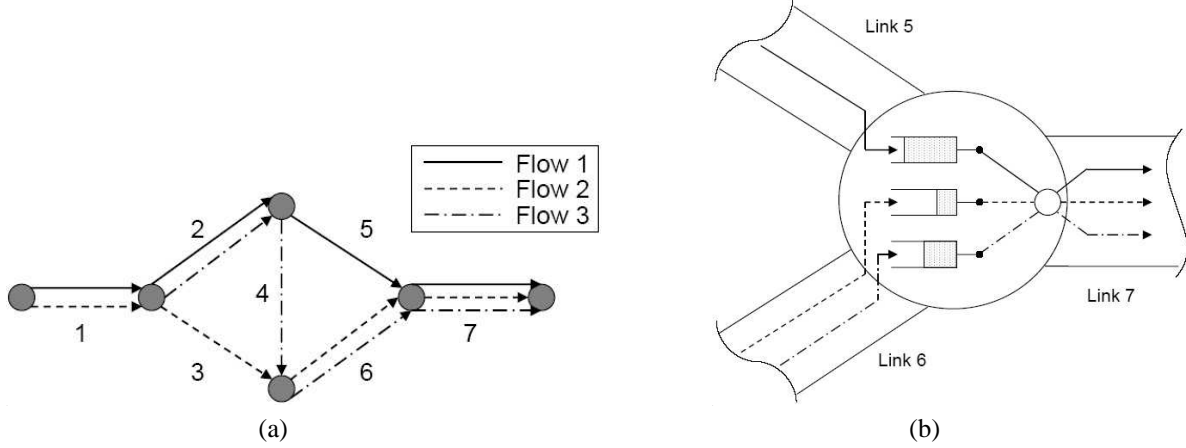\end{aligned}
$$

Figure 2. (a) Seven-link network and routes for the three flows. (b) Queues at input to link 7.

Write the entire left-hand side of the above equation as a vector $K(f, p)$, so that the KKT condition may be rewritten as $K(f, p) = 0$. Define the *KKT norm* for a pair (f, p) as $\eta(f, p) = \|K(f, p)\|$. Then, a necessary and sufficient condition for optimality of the pair (f, p) is $\eta(f, p) = 0$. We will use the KKT norm as a measure of "distance from optimality." In particular, our update algorithms converge to optimality if and only if $\eta(f(t), p(t)) \to 0$ as $t \to \infty$.

4. EMPIRICAL EVALUATION

The main goal of this section is to illustrate the operation and convergence of the algorithm. To this end, we consider a fixed network and three different scenarios on this network. These three scenarios cover the three types of joint dependence of flows in the utility function: independent, substitute, and complementary.

The network we consider is illustrated in Figure 2(a). There are $L = 7$ links. (The label on each link is its index, needed to identify it later.) We consider $N = 3$ flows taking three different routes, as shown in Figure 2(a). Notice that some links are shared while others are not. For example, link 1 is shared between flows 1 and 2, but link 3 only carries flow 2. We do not show link-capacity values here—these values are specific to each of the three scenarios, and will be given below.

Time is divided into discrete steps, and iterations in our update algorithms coincide with these steps. Flow rates are interpreted as amounts per time step, just like in any discrete-time fluid model. The local updating agents here are simply the sources of the flows (i.e., each of the three flow sources update their own flow rates).

The sharing of links is handled in the following way. At (the input to) each link we maintain one queue (buffer) for each flow sharing the link. Figure 2(b) illustrates the three queues at the input to link 7. At each time, the *input* to the queue for a flow at a link is equal to the amount of flow that traversed the *upstream* link in its respective route, which in turn is equal to the *output* of the queue for that flow at that upstream link. In the case of a first link in the route of a flow, where there is no upstream link, the input to the queue is the flow rate generated by the source of the flow at that time step. The queue evolves over time according to Lindley's equation:

$$q(t + 1) = [q(t) + (\text{input at } t) - (\text{output at } t)]_+,$$

where $q(t)$ represents the amount of data (*queue length*) in a (generic) queue at time t. Notice that at any given time, the input to the queue may exceed the output—in this case, the excess data is simply queued, contributing to an increase in the queue length.

We have already said how the input to a queue at time t is calculated. It remains to describe how the output is determined. The basic idea is that the link capacity is divided among the queues at that link (i.e., among all the flows sharing that link). Suppose link l (with capacity c_l) has three queues, with queue lengths $q_i(t)$, $i = 1, 2, 3$. We divide the capacity c in proportion to the queue lengths:

$$\text{output of queue } i \text{ on link } l \text{ at time } t = c_l \left(\frac{q_i(t)}{\sum_{j=1}^{3} q_j(t)} \right).$$

This method of calculating the link-capacity share for each flow approximates a first-in-first-out server that considers all the data from all the flows together in a single queue. Recall that the link rate is allowed to exceed the link capacity—the excess input is simply buffered in the queues, according to the sharing mechanism above.

Because of the queues at each link, the data that is transmitted from a source experiences queueing delays along the route to the destination. The average delay at a queue depends on the ratio of the rate of data arriving to the queue (input rate) to the server capacity of the queue (output rate)—this ratio is called the *load factor*. The load factor has to be less than one for the queue to be stable (i.e., for an average delay to exist). In our rate control algorithm, the goal is for each flow rate to converge to an optimal rate, which may be such that the link rate is equal to the link capacity. Therefore, we can control the asymptotic load by setting the "target" link capacity (what we called c_l earlier) to be *a fraction* of the *actual* link capacity. A typical value of this fraction in a practical network is in the range 0.2–0.7. In our experiments below, the value of this fraction is set to 0.6, which leads to significant queueing delays. This serves to illustrate the impact of delays on our rate control algorithm. Henceforth, the term "link capacity" should be taken to mean the target link capacity (represented by c_l), which is a fraction of 0.6 of the actual link capacity.

The random delays in the propagation of flows through the network because of the queues at each link are not the only sources of delays. In our experiments, we also introduce random delays to the transmission of prices and marginal utility values to the sources (rate updating agents). Specifically, the link prices used during each rate update are randomly delayed versions of the actual link prices. Similarly, the marginal utility value used for each rate update is the marginal utility evaluated at link price values from random times in the past. In our experiments, we used i.i.d. geometrically distributed delays with mean 10.

In our experiments, the parameters in the rate control algorithm were set by manual tuning—this turned out to be relatively easy. More systematic methods for tuning these controller parameters are available in the literature (e.g., Ref. 6), but a discussion of these methods is beyond our current scope.

In all scenarios below, the minimum flow rates m_i are all set to zero. The maximum flow rates M_i are taken to be sufficiently large not to matter unless otherwise specified. Similarly, except where otherwise specified, the link capacity values c_l are sufficiently large to be irrelevant. In the scenarios below, we set specific link-capacity values for certain links to determine which links are "bottlenecks."

Scenario 1: Independent flows. In this first scenario, we consider a utility function of the following form, representing the case of *independent* flows:

$$U(f_1, f_2, f_3) = u(f_1) + u(f_2) + u(f_3),$$

where $u(f) = 1 - e^{-x}$. The network link capacities are as follows: $c_4 = 0.2$, $c_7 = 1$, and, as mentioned before, all other c_l values are sufficiently large to be irrelevant. In other words, only links 2 and 7 are "bottleneck" links—we should expect only their link prices to be nonzero, and all the others zero. The only constraining maximum flow rate parameter here is $M_2 = 0.3$. Some reflection on this simple example reveals that we should expect the optimal flow rates to be as follows: flow 2 should be limited to 0.3 (because of M_2), flow 3 should be limited to 0.2 (because of c_2), and flow 1 to take up the remaining slack of link 7 (with $c_7 = 1$), so that its flow is 0.5.

Figure 3 shows the sequence of flow rates and KKT norms plotted as a function of time. Notice that the flow rates converge to the values we expected to see, based on our discussion above. Indeed, we can also see the KKT norm converging to zero, providing "certification" of convergence of the flow rates to optimality. As further verification of this optimality, we solved the global optimization problem offline using the Matlab tool `fmincon` (from Matlab's optimization toolbox). The asymptotic flow rates indeed coincide with the solution obtained using `fmincon`. Moreover, we verified that the link (and route) prices also converge to the values computed using `fmincon` (because of space limitations, we do not show plots of these prices).

Scenario 2: Substitute flows. In this scenario, we consider a joint utility function with *substitute* flows:

$$U(f_1, f_2, f_3) = u(f_1 + f_2 + f_3),$$

with u as in the Scenario 1. Our network constraint parameters are: $c_2 = c_4 = c_6 = 0.5$. An inspection of the network routes suggests that flow 3 should be turned off, while flows 1 and 2 occupy the entire link capacities of the links they share with flow 3 (links 2 and 6)—so their flow rates should both be 0.5.
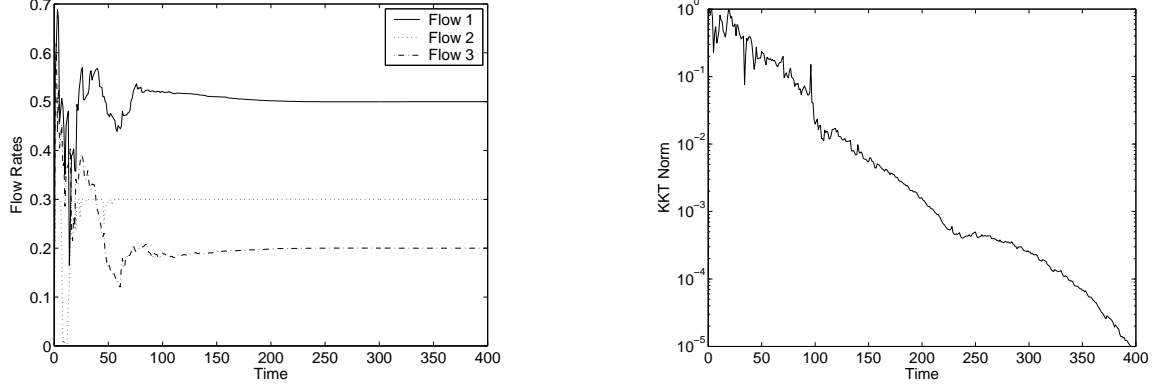
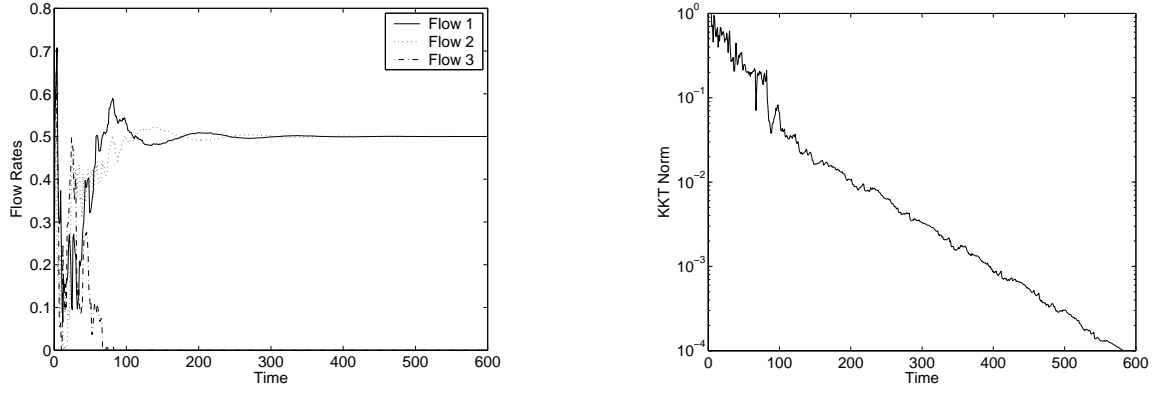**Figure 3.** Scenario 1: Flow rates and KKT norm.

Figure 4. Scenario 2: Flow rates and KKT norm.

Figure 4 shows that the algorithm behaves as expected, converging to the flow rates indicated above, and with KKT norm converging to zero. As before, our `fmincon` computation yielded the same result.

Scenario 3: Complementary flows. Finally, we consider a joint utility function with a combination of independent and *complementary* flows:

$$U(f_1, f_2, f_3) = u(f_1) + u(g(f_2, f_3)),$$

where $g(f_2, f_3) = ((f_2 + \varepsilon)^{-8} + (f_3 + \varepsilon)^{-8})^{-1/8} - (2\varepsilon^{-8})^{-1/8}$. Some explanation is in order. The above form of utility function approximates the case where flows 2 and 3 are perfect complements: $g(f_2, f_3) \approx \min(f_2, f_3)$. The perfect complementary case does not yield to a differentiable utility function. Two modifications are necessary. First, we use the approximation $\min(f_2, f_3) \approx (f_2^{-8} + f_3^{-8})^{-1/8}$. This still needs modification because it is not differentiable at 0—so we add ε to f_2 and f_3 and shift g down by a constant so that $g(0, 0) = 0$. Here, we set $\varepsilon = 10^{-3}$.

We set the link capacities as follows: $c_1 = 1$ and $c_4 = 0.4$. We immediately expect $f_3 = 0.4$. However, because of the complementary nature of flows 2 and 3, we also expect $f_2 \approx 0.4$. This leaves approximately 0.6 for flow 1.

Figure 5 shows that our rate control algorithm converges to the flow rates we expect to see, and that the KKT norm converges to zero. Once again, our `fmincon` computation yielded the same result.

Realistic tracking scenarios. Our experiments here serve primarily to illustrate that our rate control method converges to optimality. It is of interest to perform experiments with our method in realistic tracking scenarios. Specifically, in such scenarios, the utility functions are derived from tracking performance metrics, such as the trace of the error covariance.[3] Because the target(s) being tracked and the sensor platform are mobile, the utility functions are also typically time varying. In this case, the goal is not convergence to optimal rate values (which applies only in static scenarios), but adaptivity of rates to changing conditions. Because of space limitations, we are unable to provide experimental results in such a realistic setting. These will be available in a forthcoming paper.
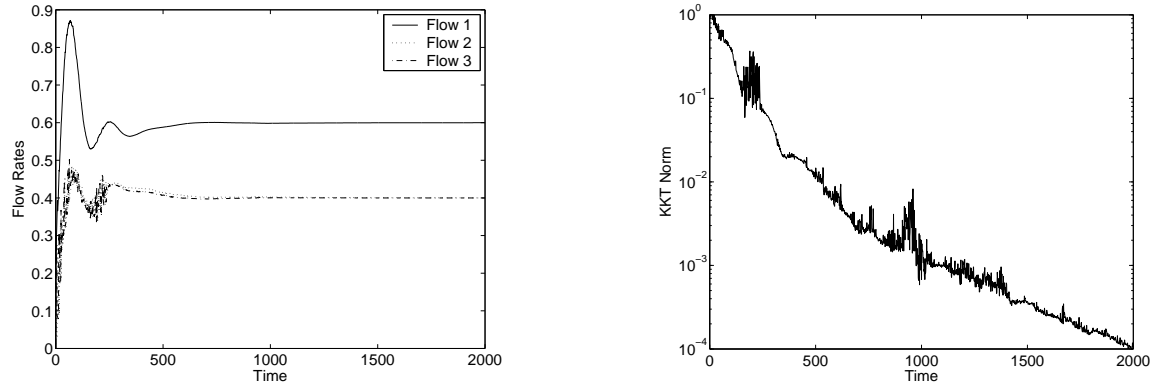
Figure 5. Scenario 3: Flow rates and KKT norm.

REFERENCES

1. S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
2. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, MA, 1997.
3. B. Brewington, F. Obermeyer, and A. Poore, "A Market-Based Framework for UAV Network Information Interchange," AFOSR Performance Report, Nov. 2004.
4. E. K. P. Chong and S. H. Żak, *An Introduction to Optimization, Second Edition*, John Wiley and Sons, New York, NY, 2001
5. R. Evans, V. Krishnamurthy, G. Nair, and L. Sciacca, "Networked sensor management and data rate control for tracking maneuvering targets," *IEEE Trans. on Signal Processing*, vol. 53, no. 6, pp. 1979–1991, June 2005.
6. G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control System Design*, Prentice Hall, Upper Saddle River, NJ, 2001.
7. F. P. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecommunications*, vol. 8, pp. 33–37, 1997.
8. F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998.
9. S. H. Low and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence," *IEEE/ACM Trans. on Networking*, vol. 7, no. 6, pp. 861–75, Dec. 1999.
10. S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Trans. on Networking*, vol. 11, no. 4, pp. 525–536, Aug. 2003.
11. R. R. Mazumdar, L. Mason and C. Douligeris, "Fairness in network optimal flow control: Optimality of product forms," *IEEE Trans. on Communications*, vol. 39, no. 5, pp. 775–782, May 1991.
12. C. Menger, *Principles of Economics*, Ludwig von Mises Institute, Auburn, AL, 2004.
13. J. Wang, L. Li, S. H. Low and J. C. Doyle, "Cross-layer Optimization in TCP/IP Networks," *IEEE/ACM Trans. on Networking*, 2005, to appear.